Universal Safety for Timed Petri Nets is PSPACE-complete

Universal Safety for Timed Petri Nets is PSPACE-complete

Parosh Aziz Abdulla
Uppsala University, Sweden

Mohamed Faouzi Atig
Uppsala University, Sweden

Radu Ciobanu
University of Edinburgh, UK

Richard Mayr
University of Edinburgh, UK

Patrick Totzke
University of Edinburgh, UK
https://orcid.org/0000-0001-5274-8190

Abstract

A timed network consists of an arbitrary number of initially identical 1-clock timed automata, interacting via hand-shake communication. In this setting there is no unique central controller, since all automata are initially identical. We consider the universal safety problem for such controller-less timed networks, i.e., verifying that a bad event (enabling some given transition) is impossible regardless of the size of the network.

This universal safety problem is dual to the existential coverability problem for timed-arc Petri nets, i.e., does there exist a number m of tokens, such that starting with m tokens in a given place, and none in the other places, some given transition is eventually enabled.

We show that these problems are PSPACE-complete.

2012 ACM Subject Classification Theory of computation → Timed and hybrid models

Keywords and phrases timed networks, safety checking, Petri nets, coverability

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.6

Funding This work was supported by the EPSRC, grant EP/M027651/1.

1 Introduction

Background. Timed-arc Petri nets (TPN) [4, 16, 3, 8, 13] are an extension of Petri nets where each token carries one real-valued clock and transitions are guarded by inequality constraints where the clock values are compared to integer bounds (via strict or non-strict inequalities). The known models differ slightly in what clock values newly created tokens can have, i.e., whether newly created tokens can inherit the clock value of some input token of the transition, or whether newly created tokens always have clock value zero. We consider the former, more general, case.

Decision problems associated with the reachability analysis of (extended) Petri nets include *Reachability* (can a given marking reach another given marking?) and *Coverability* (can a given marking ultimately enable a given transition?).

While Reachability is undecidable for all these TPN models [15], Coverability is decidable using the well-quasi ordering approach of [1, 10] and complete for the hyper-Ackermannian

complexity class F_{ω^ω} [12]. With respect to Coverability, TPN are equivalent [7] to (linearly ordered) data nets [14].

The *Existential Coverability* problem for TPN asks, for a given place p and transition t, whether there exists a number m such that the marking $M(m) \stackrel{\text{def}}{=} m \cdot \{(p, \mathbf{0})\}$ ultimately enables t. Here, $M(m)$ contains exactly m tokens on place p with all clocks set to zero and *no other tokens*. This problem corresponds to checking safety properties in distributed networks of arbitrarily many (namely m) initially identical timed processes that communicate by handshake. A negative answer certifies that the 'bad event' of transition t can never happen regardless of the number m of processes, i.e., the network is safe for any size. Thus by checking existential coverability, one solves the dual problem of *Universal Safety*. (Note that the m timed tokens/processes are only initially identical. They can develop differently due to non-determinacy in the transitions.)

The corresponding problem for timed networks studied in [2] does not allow the dynamic creation of new timed processes (unlike the TPN model which can increase the number of timed tokens), but considers multiple clocks per process (unlike our TPN with one clock per token).

The TPN model above corresponds to a distributed network without a central controller, since initially there are no tokens on other places that could be used to simulate one. Adding a central controller would make *Existential Coverability* polynomially inter-reducible with normal *Coverability* and thus complete for F_{ω^ω} [12] (and even undecidable for > 1 clocks per token [2]).

Aminof et. al. [6] study the model checking problem of ω-regular properties for the controller-less model and in particular claim an EXPSPACE upper bound for checking universal safety. However, their result only holds for discrete time (integer-valued clocks) and they do not provide a matching lower bound.

Our contribution. We show that *Existential Coverability* (and thus universal safety) is decidable and PSPACE-complete. This positively resolves an open question from [2] regarding the decidability of universal safety in the controller-less networks. Moreover, a symbolic representation of the set of coverable configurations can be computed (using exponential space).

The PSPACE lower bound is shown by a reduction from the iterated monotone Boolean circuit problem. (It does not follow directly from the PSPACE-completeness of the reachability problem in timed automata of [5], due to the lack of a central controller.)

The main ideas for the PSPACE upper bound are as follows. First we provide a logspace reduction of the Existential Coverability problem for TPN to the corresponding problem for a syntactic subclass, non-consuming TPN. Then we perform an abstraction of the real-valued clocks, similar to the one used in [3]. Clock values are split into integer parts and fractional parts. The integer parts of the clocks can be abstracted into a finite domain, since the transition guards cannot distinguish between values above the maximal constant that appears in the system. The fractional parts of the clock values that occur in a marking are ordered sequentially. Then every marking can be abstracted into a string where all the tokens with the i-th fractional clock value are encoded in the i-th symbol in the string. Since token multiplicities do not matter for existential coverability, the alphabet from which these strings are built is finite. The primary difficulty is that the length of these strings can grow dynamically as the system evolves, i.e., the space of these strings is still infinite for a given TPN. We perform a forward exploration of the space of reachable strings. By using an acceleration technique, we can effectively construct a symbolic representation of the

set of reachable strings in terms of finitely many regular expressions. Finally, we can check existential coverability by using this symbolic representation.

2 Timed Petri Nets

We use $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ to denote the sets of nonnegative integers and reals, respectively. For $n \in \mathbb{N}$ we write $[n]$ for the set $\{0, \ldots, n\}$.

For a set A, we use A^* to denote the set of words, i.e. finite sequences, over A, and write ε for the empty word. If R is a regular expression over A then $\mathcal{L}(R) \subseteq A^*$ denotes its language.

A *multiset* over a set X is a function $M : X \to \mathbb{N}$. The set $X^{\oplus}$ of all (finitely supported) multisets over X is partially ordered pointwise (by $\leq$). The multiset union of $M, M' \in X^{\oplus}$ is $(M \oplus M') \in X^{\oplus}$ with $(M \oplus M')(\alpha) \stackrel{\text{def}}{=} M(\alpha) + M'(\alpha)$ for all $\alpha \in X$. If $M \geq M'$ then the multiset difference $(M \ominus M')$ is the unique $M'' \in X^{\oplus}$ with $M = M' \oplus M''$. We will use a monomial representation and write for example $(\alpha + \beta^3)$ for the multiset $(\alpha \mapsto 1, \beta \mapsto 3)$. For a multiset M and a number $m \in \mathbb{N}$ we let $m \cdot M$ denote the m-fold multiset sum of M. We further lift this to sets of numbers and multisets on the obvious fashion, so that in particular $\mathbb{N} \cdot S \stackrel{\text{def}}{=} \{n \cdot M \mid n \in \mathbb{N}, M \in S\}$.

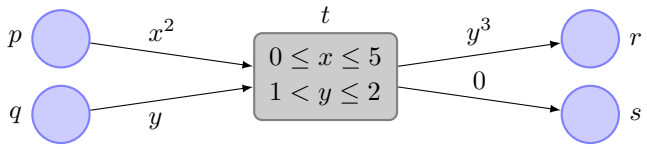
Timed Petri nets are place/transition nets where each token carries a real value, sometimes called its *clock value* or *age*. Transition firing depends on there being sufficiently many tokens whose value is in a specified interval. All tokens produced by a transition either have age 0, or inherit the age of an input-token of the transition. To model time passing, all token ages can advance simultaneously by the same (real-valued) amount.

▶ **Definition 1** (TPN). A *timed Petri net* (TPN) $\mathcal{N} = (P, T, Var, G, Pre, Post)$ consists of finite sets of *places* P, *transitions* T and *variables* Var, as well as functions $G, Pre, Post$ defining transition *guards*, *pre–* and *postconditions*, as follows.

For every transition $t \in T$, the guard $G(t)$ maps variables to (open, half-open or closed) intervals with endpoints in $\mathbb{N} \cup \{\infty\}$, restricting which values variables may take. All numbers are encoded in unary. The precondition $Pre(t)$ is a finite multiset over $(P \times Var)$. Let $Var(t) \subseteq Var$ be the subset of variables appearing positively in $Pre(t)$. The postcondition $Post(t)$ is then a finite multiset over $(P \times (\{0\} \cup Var(t)))$, specifying the locations and clock values of produced tokens. Here, the symbolic clock value is either 0 (demanding a reset to age 0), or a variable that appeared already in the precondition.

A *marking* is a finite multiset over $P \times \mathbb{R}_{\geq 0}$.

▶ **Example 2.** The picture below shows a place/transition representation of an TPN with four places and one transition. $Var(t) = \{x, y\}$, $Pre(t) = (p, x)^2 + (q, y)$, $G(t)(x) = [0, 5]$, $G(t)(y) =]1, 2]$ and $Post(t) = (r, y)^3 + (s, 0)$.



The transition t consumes two tokens from place p, both of which have the same clock value x (where $0 \leq x \leq 5$) and one token from place q with clock value y (where $1 < y \leq 2$). It produces three tokens on place r who all have the same clock value y (where y comes from the clock value of the token read from q), and another token with value 0 on place s.

There are two different binary step relations on markings: *discrete* steps $\longrightarrow_t$ which fire a transition t as specified by the relations G, Pre, and $Post$, and *time passing* steps $\longrightarrow_d$ for durations $d \in \mathbb{R}_{\geq 0}$, which simply increment all clocks by d.

▶ **Definition 3** (Discrete Steps). For a transition $t \in T$ and a variable evaluation $\pi : Var \to \mathbb{R}_{\geq 0}$, we say that π *satisfies* $G(t)$ if $\pi(x) \in G(t)(x)$ holds for all $x \in Var$. By lifting π to multisets over $(P \times Var)$ (respectively, to multisets over $(P \times (\{0\} \cup Var))$ with $\pi(0) = 0$) in the canonical way, such an evaluation translates preconditions $Pre(t)$ and $Post(t)$ into markings $\pi(Pre(t))$ and $\pi(Post(t))$, where for all $p \in P$ and $c \in \mathbb{R}_{\geq 0}$,

$$\pi(Pre(t))(p,c) \stackrel{\text{def}}{=} \sum_{\pi(v)=c} Pre(t)(p,v) \qquad \text{and} \qquad \pi(Post(t))(p,c) \stackrel{\text{def}}{=} \sum_{\pi(v)=c} Post(t)(p,v).$$

A transition $t \in T$ is called *enabled* in marking M, if there exists an evaluation π that satisfies $G(t)$ and such that $\pi(Pre(t)) \leq M$. In this case, there is a discrete step $M \longrightarrow_t M'$ from marking M to M', defined as $M' = M \ominus \pi(Pre(t)) \oplus \pi(Post(t))$.

▶ **Definition 4** (Time Steps). Let M be a marking and $d \in \mathbb{R}_{\geq 0}$. There is a time step $M \longrightarrow_d M'$ to the marking M' with $M'(p,c) \stackrel{\text{def}}{=} M(p, c-d)$ for $c \geq d$, and $M'(p,c) \stackrel{\text{def}}{=} 0$, otherwise. We also refer to M' as $(M + d)$.

We write $\to_{Time}$ for the union of all timed steps, $\to_{Disc}$ for the union of all discrete steps and simply $\to$ for $\to_{Disc} \cup \to_{Time}$. The transitive and reflexive closure of $\to$ is $\stackrel{*}{\to}$. $Cover(M)$ denotes the set of markings M' for which there is an $M'' \geq M'$ with $M \stackrel{*}{\to} M''$.

We are interested in the *existential coverability problem* (∃COVER for short), as follows.

Input: A TPN, an initial place p and a transition t.
Question: Does there exist $M \in Cover(\mathbb{N} \cdot \{(p,0)\})$ that enables t?

We show that this problem is PSPACE-complete. Both lower and upper bound will be shown (w.l.o.g., see Lemma 8) for the syntactic subclass of *non-consuming* TPN, defined as follows.

▶ **Definition 5.** A *timed Petri net* $(P, T, Var, G, Pre, Post)$ is *non-consuming* if for all $t \in T$, $p \in P$ and $x \in Var$ it holds that both 1) $Pre(t)(p,x) \leq 1$, and 2) $Pre(t) \leq Post(t)$.

In a non-consuming TPN, token multiplicities are irrelevant for discrete transitions. Intuitively, having one token (p,c) is equivalent to having an inexhaustible supply of such tokens.

The first condition is merely syntactic convenience. It asks that each transition takes at most one token from each place. The second condition in Definition 5 implies that for each discrete step $M \longrightarrow_t M'$ we have $M' \geq M$. Therefore, once a token (p,c) is present on a place p, it will stay there unchanged (unless time passes), and it will enable transitions with (p,c) in their precondition.

Wherever possible, we will from now on therefore allow ourselves to use the set notation for markings, that is simply treat markings $M \in (P \times \mathbb{R}_{\geq 0})^{\oplus}$ as sets $M \subseteq (P \times \mathbb{R}_{\geq 0})$.

3 Lower Bound

PSPACE-hardness of ∃COVER does not follow directly from the PSPACE-completeness of the reachability problem in timed automata of [5]. The non-consuming property of our TPN makes it impossible to fully implement the control-state of a timed automaton. Instead our

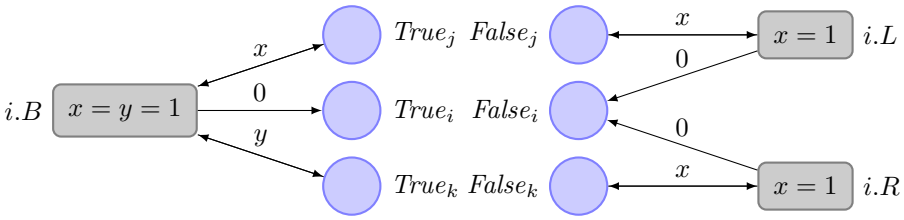


Figure 1 The transitions $i.B, i.R$ and $i.L$ that simulate the update of bit i according to constraint $i' = j \wedge k$. All transitions demand that incoming tokens are of age exactly 1 and only tokens of age 0 are produced.

proof uses multiple timed tokens and a reduction from the iterated monotone Boolean circuit problem [11].

A depth-1 monotone Boolean circuit is a function $F : \{0,1\}^n \to \{0,1\}^n$ represented by n constraints: For every $0 \leq i < n$ there is a constraint of the form $i' = j \otimes k$, where $0 \leq j, k < n$ and $\otimes \in \{\wedge, \vee\}$, which expresses how the next value of bit i depends on the current values of bits j and k. For every bitvector $\boldsymbol{v} \in \{0,1\}^n$, the function F then satisfies $F(\boldsymbol{v})[i] \stackrel{\text{def}}{=} \boldsymbol{v}[j] \otimes \boldsymbol{v}[k]$. It is PSPACE-complete to check whether for a given vector $\boldsymbol{v} \in \{0,1\}^n$ there exists a number $m \in \mathbb{N}$ such that $F^m(\boldsymbol{v})[0] = 1$.

Towards a lower bound for ∃COVER (Theorem 7) we construct a non-consuming TPN as follows, for a given circuit. The main idea is to simulate circuit constraints by transitions that reset tokens of age 1 (encoding $\boldsymbol{v}$) to fresh ones of age 0 (encoding $F(\boldsymbol{v})$), and let time pass by one unit to enter the next round.

For every bit $0 \leq i < n$, the net contains two places $True_i$ and $False_i$. A marking $M_{\boldsymbol{v}} \leq P \times \mathbb{R}_{\geq 0}$ is an *encoding* of a vector $\boldsymbol{v} \in \{0,1\}^n$ if for every $0 \leq i < n$ the following hold.

1. $(True_i, 0) \in M_{\boldsymbol{v}} \iff \boldsymbol{v}[i] = 1$.
2. $(False_i, 0) \in M_{\boldsymbol{v}} \iff \boldsymbol{v}[i] = 0$.
3. If $(p, c) \in M_{\boldsymbol{v}}$ then $c = 0$ or $c \geq 1$.

Note that in particular one cannot have both $(True_i, 0)$ and $(False_i, 0)$ in $M_{\boldsymbol{v}}$. For every constraint $i' = j \wedge k$ we introduce three transitions, $i.L, i.R$, and $i.B$, where

$$Pre(i.B) \stackrel{\text{def}}{=} (True_j, x) + (True_k, y) \qquad Post(i.B) \stackrel{\text{def}}{=} Pre(i.B) + (True_i, 0)$$
$$Pre(i.L) \stackrel{\text{def}}{=} (False_j, x) \qquad Post(i.L) \stackrel{\text{def}}{=} Pre(i.L) + (False_i, 0)$$
$$Pre(i.R) \stackrel{\text{def}}{=} (False_k, x) \qquad Post(i.R) \stackrel{\text{def}}{=} Pre(i.R) + (False_i, 0)$$

and the guard for all transitions is $G(x) = G(y) = 1$. See Figure 1 for an illustration. For disjunctions $i' = j \vee k$ the transitions are defined analogously, with *True* and *False* inverted. The correctness proof of our construction rests on the following simple observation.

▶ **Lemma 6.** *If $F(\boldsymbol{v}) = \boldsymbol{v}'$ then for every encoding $M_{\boldsymbol{v}}$ of $\boldsymbol{v}$, there exists an encoding $M_{\boldsymbol{v}'}$ of $\boldsymbol{v}'$ such that $M_{\boldsymbol{v}} \longrightarrow_1 \stackrel{*}{\to}_{Disc} M_{\boldsymbol{v}'}$. Conversely, if $M_{\boldsymbol{v}} \longrightarrow_1 \stackrel{*}{\to}_{Disc} M_{\boldsymbol{v}'}$ for encodings $M_{\boldsymbol{v}}$ and $M_{\boldsymbol{v}'}$ of $\boldsymbol{v}$ and $\boldsymbol{v}'$ respectively, then $F(\boldsymbol{v}) = \boldsymbol{v}'$.*

Proof. For the first part, we construct a sequence $M_0 \to_{Disc} M_1 \to_{Disc} \ldots \to_{Disc} M_{n-1}$ where $M_0 \stackrel{\text{def}}{=} (M_{\boldsymbol{v}} + 1)$ and every step $M_{i-1} \to_{Disc} M_i$ adds tokens simulating the ith constraint of F. Since the TPN is non-consuming, we will have that $M_i \geq (M_{\boldsymbol{v}} + 1)$, for all $i < n$. Consider now constraint i', and assume w.l.o.g. that $i' = j \wedge k$ (the other case is analogous). There are two cases depending on $\boldsymbol{v}'[i]$.

1. Case $\boldsymbol{v'}[i] = 1$. By our assumption that $F(\boldsymbol{v}) = \boldsymbol{v'}$ we know that $\boldsymbol{v}[j] = 1$ and $\boldsymbol{v}[k] = 1$. So $(True_j, 1) \in (M_{\boldsymbol{v}} + 1) \leq M_{i-1}$ and $(True_k, 1) \in (M_{\boldsymbol{v}} + 1) \leq M_{i-1}$. By construction of the net, there is a transition $i.B$ with $Pre(i.B) = (True_j, 1) + (True_k, 1)$ and $Post(i.B) = Pre(i.B) + (True_i, 0)$. This justifies step $M_{i-1} \longrightarrow_{i.B} M_i$ and therefore that $(True_i, 0) \in M_i \leq M_{n-1}$. Also notice that no marking reachable from M_0 using only discrete steps can contain the token $(False_i, 0)$. This is because these can only be produced by transitions requiring either $(False_j, 1)$ or $(False_k, 1)$, which are not contained in M_0 by assumption that $M_{\boldsymbol{v}}$ encodes $\boldsymbol{v}$. Therefore $(False_i, 0) \notin M_{n-1}$.
2. Case $\boldsymbol{v'}[i] = 0$. W.l.o.g., $\boldsymbol{v}[j] = 0$. Therefore, $(False_j, 1) \in (M_{\boldsymbol{v}} + 1) \leq M_{i-1}$. By construction of the net, there exists transition $i.L$ with $Pre(i.L) = (False_j, 1)$ and $Post(i.L) = Pre(i.L) + (False_i, 0)$. This justifies the step $M_{i-1} \longrightarrow_{i.L} M_i$, with $(False_i, 0) \in M_i \leq M_{n-1}$. Notice again that no marking reachable from M_0 using only discrete steps can contain the token $(True_i, 0)$. This is because these can only be produced by transitions $i.B$, requiring both $(True_j, 1), (True_k, 1) \in M_0$, contradicting our assumptions. Hence, $(True_i, 0) \notin M_{n-1}$.

We conclude that the constructed marking M_{n-1} is an encoding of $\boldsymbol{v'}$.

For the other part of the claim, assume that there exist markings $M_{\boldsymbol{v}}$ and $M_{\boldsymbol{v'}}$ which are encodings of vectors $\boldsymbol{v}$ and $\boldsymbol{v'}$, respectively, with $M_{\boldsymbol{v}} \longrightarrow_1 \xrightarrow{*}_{Disc} M_{\boldsymbol{v'}}$. We will show that $F(\boldsymbol{v}) = \boldsymbol{v}'$. Recall that $F(\boldsymbol{v})[i] \stackrel{\text{def}}{=} \boldsymbol{v}[j] \otimes \boldsymbol{v}[k]$, where $0 \leq j, k < n$ and $\otimes \in \{\wedge, \vee\}$. We will show for each $i < n$ that $\boldsymbol{v}'[i] = \boldsymbol{v}[j] \otimes \boldsymbol{v}[k]$. Again, consider the constraint i', and assume w.l.o.g. that $i' = j \wedge k$ (the other case is analogous). There are two cases.

1. Case $\boldsymbol{v'}[i] = 1$. By definition of a marking encoding, we have that $(True_i, 0) \in M_{\boldsymbol{v}}$. By construction, there is a transition $i.B$ with $Pre(i.B) = (True_j, 1) + (True_k, 1)$ and $Post(i.B) = Pre(i.B) + (True_i, 0)$. By assumption, it holds that $(M_{\boldsymbol{v}} + 1) \xrightarrow{*}_{Disc} M'_{\boldsymbol{v}}$, where $M_{\boldsymbol{v}} \longrightarrow_1 (M_{\boldsymbol{v}} + 1)$. Note that $(True_j, 1) \in (M_{\boldsymbol{v}} + 1)$ and $(True_k, 1) \in (M_{\boldsymbol{v}} + 1)$. Hence, we have that $\boldsymbol{v}[j] = 1$ and $\boldsymbol{v}[k] = 1$, and therefore that $F(\boldsymbol{v})[i] = \boldsymbol{v'}[i] = \boldsymbol{v}[j] \wedge \boldsymbol{v}[k]$.
2. Case $\boldsymbol{v'}[i] = 0$. Then $(False_i, 0) \in M_{\boldsymbol{v}}$ and, since this token can only be produced by transitions $i.L$ or $i.R$, either $(False_j, 1) \in (M_{\boldsymbol{v}} + 1)$ or $(False_k, 1) \in (M_{\boldsymbol{v}} + 1)$. Therefore $(False_j, 0) \in (M_{\boldsymbol{v}})$ or $(False_k, 0) \in (M_{\boldsymbol{v}})$ and because $M_{\boldsymbol{v}}$ is an encoding of $\boldsymbol{v}$, this means that either $\boldsymbol{v}[j] = 0$ or $\boldsymbol{v}[k] = 0$. Therefore, $F(\boldsymbol{v'})[i] = \boldsymbol{v}[j] \wedge \boldsymbol{v}[k] = 0$. ◀

▶ **Theorem 7.** *$\exists$COVER is* PSPACE*-hard for non-consuming TPN.*

Proof. For a given monotone Boolean circuit, define a non-consuming TPN as above. By induction on $m \in \mathbb{N}$ using Lemma 6, we derive that there exists $m \in \mathbb{N}$ with $F^m(\boldsymbol{v}) = \boldsymbol{v'}$ and $\boldsymbol{v}'[0] = 1$ if, and only if, there exists encodings $M_{\boldsymbol{v}}$ of $\boldsymbol{v}$ and $M_{\boldsymbol{v'}}$ of $\boldsymbol{v'}$, with $M_{\boldsymbol{v}} \xrightarrow{*} M_{\boldsymbol{v}'}$. Moreover, if there is a marking M such that $M_{\boldsymbol{v}} \xrightarrow{*} M$ and $0 \in frac(M)$, where M contains a token of age 0, then $M \leq M_{\boldsymbol{v'}}$ for some encoding $M_{\boldsymbol{v}'}$ of a vector $\boldsymbol{v'} = F^m(\boldsymbol{v})$. This means that it suffices to add one transition t with $Pre(t) = (True_0, 0)$ whose enabledness witnesses the existence of a reachable encoding $M_{\boldsymbol{v}'}$ containing a token $(True_0, 0)$. By the properties above, there exists $m \in \mathbb{N}$ with $F^m(\boldsymbol{v}) = \boldsymbol{v'}$ and $\boldsymbol{v}'[0] = 1$ iff $M_{\boldsymbol{v}} \xrightarrow{*} M_{\boldsymbol{v}'} \xrightarrow{t}$. ◀

This lower bound holds even for discrete time TPN, e.g. [9], because the proof uses only timed steps with duration $d = 1$.

4 Upper Bound

We start by observing that we can restrict ourselves, without loss of generality, to non-consuming TPN (Definition 5) for showing the upper bound. Intuitively, since we start with

an arbitrarily high number of tokens anyway, it does not matter how many of them are consumed by transitions during the computation, since some always remain.

▶ **Lemma 8.** *The ∃COVER problem for TPN logspace-reduces to the ∃COVER problem for non-consuming TPN. That is, for every TPN $\mathcal{N}$ and for every place p and transition t of $\mathcal{N}$, one can construct, using logarithmic space, a non-consumimg TPN $\mathcal{N}'$ together with a place p' and transition t' of $\mathcal{N}'$, so that there exists $M \in Cover_{\mathcal{N}}(\mathbb{N} \cdot \{(p,0)\})$ enabling t in $\mathcal{N}$ if and only if there exists $M' \in Cover_{\mathcal{N}'}(\mathbb{N} \cdot \{(p',0)\})$ that enables t' in $\mathcal{N}'$.*

4.1 Region Abstraction

We recall a constraint system called regions defined for timed automata [5]. The version for TPN used here is similar to the one in [3].

Consider a fixed, nonconsuming TPN $\mathcal{N} = (P, T, Var, G, Pre, Post)$. Let c_{max} be the largest finite value appearing in transition guards G. Since different tokens with age $> c_{max}$ cannot be distinguished by transition guards, we consider only token ages below or equal to c_{max} and treat the integer parts of older tokens as equal to $c_{max}+1$. Let $int(c) \stackrel{\text{def}}{=} \min\{c_{max}+1, \lfloor c \rfloor\}$ and $frac(c) \stackrel{\text{def}}{=} c - \lfloor c \rfloor$ for a real value $c \in \mathbb{R}_{\geq 0}$. We will work with an abstraction of TPN markings as words over the alphabet $\Sigma \stackrel{\text{def}}{=} 2^{P \times [c_{max}+1]}$. Each symbol $X \in \Sigma$ represents the places and integer ages of tokens for a particular fractional value.

▶ **Definition 9.** Let $M \subseteq P \times \mathbb{R}_{\geq 0}$ be a marking and let $frac(M) \stackrel{\text{def}}{=} \{frac(c) \mid (p,c) \in M\}$ be the set of fractional clock values that appear in M.

Let $S \subset [0,1[$ be a finite set of real numbers with $0 \in S$ and $frac(M) \subseteq S$ and let $f_0, f_1, \ldots, f_n$, be an enumeration of S so that $f_{i-1} < f_i$ for all $i \leq n$. The *S-abstraction* of M is

$$abs_S(M) \stackrel{\text{def}}{=} x_0 x_1 \ldots x_n \in \Sigma^*$$

where $x_i \stackrel{\text{def}}{=} \{(p, int(c)) \mid (p,c) \in M \wedge frac(c) = f_i\}$ for all $i \leq n$. We simply write $abs(M)$ for the shortest abstraction, i.e. with respect to $S = \{0\} \cup frac(M)$.

▶ **Example 10.** The abstraction of marking $M = \{(p,2.1),(q,2.2),(p,5.1),(q,5.1)\}$ is $abs(M) = \emptyset\ \{(p,2),(p,5),(q,5)\}\ \{(q,2)\}$. The first symbol is $\emptyset$, because M contains no token with an integer age (i.e., no token whose age has fractional part 0). The second and third symbols represent sets of tokens with fractional values 0.1 and 0.2, respectively.

Clocks with integer values play a special role in the behavior of TPN, because the constants in the transition guards are integers. Thus we always include the fractional part 0 in the set S in Definition 9.

We use a special kind of regular expressions over Σ to represent coverable sets of TPN markings as follows.

▶ **Definition 11.** A regular expression E over Σ represents the downward-closed set of TPN markings covered by one that has an abstraction in the language of E:

$$[\![E]\!] \stackrel{\text{def}}{=} \{N \mid \exists M \exists S.\ M \geq N \wedge abs_S(M) \in \mathcal{L}(E)\}.$$

An expression is *simple* if it is of the form $E = x_0 x_1 \ldots x_k$ where for all $i \leq k$ either $x_i \in \Sigma$ or $x_i = y_i^*$ for some $y_i \in \Sigma$. In the latter case we say that x_i *carries a star*. That is, a simple expression is free of Boolean combinators and uses only concatenation and Kleene star. We will write $\hat{x}_i$ to denote the symbol in Σ at position i: it is x_i if $x_i \in \Sigma$ and y_i otherwise.

▶ Remark 12. Notice that for all simple expressions α, β so that $|\alpha| > 0$, we have that $[\![\alpha\emptyset\beta]\!] = [\![\alpha\beta]\!]$. However, unless α has length 0 or is of the form $\alpha = \emptyset\alpha'$, we have $[\![\emptyset\alpha]\!] \neq [\![\alpha]\!]$. This is because a marking M that contains a token (p, c) with $frac(c) = 0$ has the property that all abstractions $abs_S(M) = x_0 \dots x_k$ of M have $x_0 \neq \emptyset$.

The following lemmas express the effect of TPN transitions at the level of the region abstraction. Lemmas 13 and 15 state that maximally firing of discrete transitions (the relation $\xrightarrow{*}_{Disc}$) is computable and monotone. Lemmas 16 and 17 state how to represent timed-step successor markings.

▶ **Lemma 13.** *For every non-consuming TPN $\mathcal{N}$ there are polynomial time computable functions $f : \Sigma \times \Sigma \times \Sigma \to \Sigma$ and $g : \Sigma \times \Sigma \times \Sigma \to \Sigma$ with the following properties.*

1. *f and g are monotone (w.r.t. subset ordering) in each argument.*
2. *$f(\alpha, \beta, x) \supseteq x$ and $g(\alpha, \beta, x) \supseteq x$ for all $\alpha, \beta, x \in \Sigma$.*
3. *Suppose that $E = x_0 x_1 \dots x_k$ is a simple expression, $\alpha \stackrel{def}{=} x_0$ and $\beta \stackrel{def}{=} \bigcup_{i>0} \hat{x}_i$, and $E' = x'_0 x'_1 \dots x'_k$ is the derived expression defined by conditions:*
   - a. *$x'_0 \stackrel{def}{=} f(\alpha, \beta, x_0)$,*
   - b. *$x'_i \stackrel{def}{=} g(\alpha, \beta, \hat{x}_i)^*$ for $i > 0$,*
   - c. *x'_i carries a star iff x_i does.*

   Then $[\![E']\!] = \{M'' \mid \exists M \in [\![E]\!] \wedge M \xrightarrow{*}_{Disc} M' \geq M''\}$.

▶ **Definition 14.** We will write $SAT(E) \stackrel{\text{def}}{=} E'$ for the successor expression E' of E guaranteed by Lemma 13. I.e., $SAT(E)$ is the saturation of E by maximally firing discrete transitions.

Notice that by definition it holds that $[\![E]\!] \subseteq [\![SAT(E)]\!] \subseteq Cover([\![E]\!])$, and consequently also that $Cover([\![SAT(E)]\!]) = Cover([\![E]\!])$.

▶ **Lemma 15.** *Suppose that $X = x_0 x_1 \dots x_k$ is a simple expression of length $k+1$ with $SAT(X) = x'_0 x'_1 \dots x'_k$ and $x_0, x'_0 \in \Sigma$. Let $Y = y_0 \alpha_1 y_1 \alpha_2 \dots \alpha_k y_k$ be a simple expression with $SAT(Y) = y'_0 \alpha'_1 y'_1 \alpha'_2 \dots \alpha'_k y'_k$ and $y_0, y'_0 \in \Sigma$.*

If $\hat{x}_i \subseteq \hat{y}_i$ for all $i \leq k$ then $\hat{x}'_i \subseteq \hat{y}'_i$ for all $i \leq k$.

Proof. The assumption of the lemma provides that $\alpha_x \stackrel{\text{def}}{=} x_0 \subseteq \alpha_y \stackrel{\text{def}}{=} y_0$ and $\beta_x \stackrel{\text{def}}{=} \bigcup_{k \geq i > 0} \hat{x}_i \subseteq \beta_y \stackrel{\text{def}}{=} \bigcup_{k \geq i > 0} \hat{y}_i$. Therefore, by Item 1 of Lemma 13, we get that

$$x'_0 = f(\alpha_x, \beta_x, x_0) \quad \subseteq \quad f(\alpha_y, \beta_y, y_0) = y'_0$$

and similarly, for all $k \geq i \geq 0$, that $\hat{x}'_i = g(\alpha_x, \beta_x, \hat{x}_i) \ \subseteq \ g(\alpha_y, \beta_y, \hat{y}_i) = \hat{y}'_i$. ◀

For $x \in \Sigma$ we write $(x+1) \stackrel{\text{def}}{=} \{(p, int(n+1)) \mid (p, n) \in x\}$ for the symbol where token ages are incremented by 1.

▶ **Lemma 16.** $[\![\emptyset E]\!] = \{M' \mid \exists M \in [\![E]\!] \wedge M \longrightarrow_d M' \wedge d < 1 - \max(frac(M))\}$.

▶ **Lemma 17.** *Let αz be a simple expression where $\hat{z} = z \in \Sigma$ (the rightmost symbol is not starred). Then, $[\![(z+1)\alpha]\!]$ contains a marking N if, and only if, there exists markings $N' \geq N$ and M, and a set $S \subseteq [0, 1[$ so that*

1. *$|S| = |\alpha z|$*
2. *$abs_S(M) \in \mathcal{L}(\alpha z)$*
3. *$M \longrightarrow_d N'$ for $d = 1 - \max(S)$.*

Proof. Suppose markings N, N', M, a set $S \subseteq [0,1[$ and $d \in \mathbb{R}_{\geq 0}$ so that the conditions 1 to 3 are satisfied. Let $S' \stackrel{\text{def}}{=} \{0\} \cup \{s+d \mid s \in S \setminus \{d\}\}$. Then, $|S'| = |S|$ and $abs_{S'}(N') \in \mathcal{L}((z+1)\alpha)$, which witnesses that $N \in [\![(z+1)\alpha]\!]$.

Conversely, let $N \in [\![(z+1)\alpha]\!]$ be a non-empty marking. If $|\alpha| = 0$, then $N \in [\![(z+1)]\!]$ and so $abs_S(N) \in \mathcal{L}((z+1))$ for $S \stackrel{\text{def}}{=} frac(N) = \{0\}$. This means that $M \longrightarrow_1 N = (M+1)$ for a marking M with $abs_S(M) \in \mathcal{L}(z) = \mathcal{L}(\alpha z)$.

If $|\alpha| > 0$, pick some marking $N' \geq N$ and set S' so that $abs_{S'}(N') = (z+1)w$, for some word $w \in \mathcal{L}(\alpha)$. Then we must have that $|S'| = |(z+1)\alpha| > 1$ and so $d \stackrel{\text{def}}{=} \min(S' \setminus \{0\})$ exists. Let $S \stackrel{\text{def}}{=} \{s-d \mid s \in S'\} \cup \{1-d\}$ and M be the unique marking with $M \longrightarrow_d N'$. Notice that $1-d = \max(S)$. It follows that $abs_S(M) = wz \in \mathcal{L}(\alpha z)$. ◀

We will often use the following simple fact, which is a direct consequence of Lemma 17.

▶ **Corollary 18.** $[\![(z+1)\alpha]\!] \subseteq Cover([\![\alpha z]\!])$.

Finally, the following lemma will be the basis for our exploration algorithm.

▶ **Lemma 19.** *Let αx_0^* be a simple expression with $SAT(\alpha x_0^*) = \alpha x_0^*$. Then $Cover([\![\alpha x_0^*]\!]) = [\![\alpha x_0^*]\!] \cup Cover([\![(x_0+1)\alpha x_0^*]\!])$.*

Proof. For the right to left inclusion notice that $[\![\alpha x_0^*]\!] \subseteq Cover([\![\alpha x_0^*]\!])$ trivially holds. For the rest, we have $[\![(x_0+1)\alpha x_0^*]\!] \subseteq Cover([\![\alpha x_0^*]\!])$ by Corollary 18, and therefore $Cover([\![(x_0+1)\alpha x_0^*]\!]) \subseteq Cover(Cover([\![\alpha x_0^*]\!])) = Cover([\![\alpha x_0^*]\!])$. For the left to right inclusion, we equivalently show that

$$Cover([\![\alpha x_0^*]\!]) \setminus [\![\alpha x_0^*]\!] \subseteq Cover([\![(x_0+1)\alpha x_0^*]\!]) \tag{1}$$

Using the assumption that $SAT(\alpha x_0^*) = \alpha x_0^*$, the set on the left contains everything coverable from $[\![\alpha x_0^*]\!]$ by a sequence that starts with a (short) time step. It can therefore be written as

$$Cover(\{N_1 \mid \exists N_0 \in [\![\alpha x_0^*]\!] \wedge N_0 \longrightarrow_d N_1 \wedge 0 < d < 1 - \max(frac(N_0))\}).$$

By Lemma 16 and because $[\![\emptyset\alpha]\!] \subseteq [\![X\alpha]\!]$ for all $X \in \Sigma$ and $\alpha \in \Sigma^*$, we conclude that indeed, $Cover([\![\alpha x_0^*]\!]) \setminus [\![\alpha x_0^*]\!] \subseteq Cover([\![\emptyset\alpha x_0^*]\!]) \subseteq Cover([\![(x_0+1)\alpha x_0^*]\!])$. ◀

4.2 Acceleration

We propose an acceleration procedure based on unfolding expressions according to Lemma 19 (interleaved with saturation steps to guarantee its premise) and introducing new Kleene stars to keep the length of intermediate expressions bounded. This procedure (depicted in Algorithm 1), is used to characterize an initial subset of the coverability set.

Algorithm 1 Accelerate

Input: a simple expression $S_0 = x_1 x_0^*$ (of length 2 and with last symbol starred)

Output: simple expressions S_1, S_i and R, of lengths 2, 4, and 2, respectively.

1: $S_1 \stackrel{\text{def}}{=} x_1^1 (x_0^1)^* = SAT(x_1 x_0^*)$
2: $S_2 \stackrel{\text{def}}{=} x_2^2 x_1^2 (x_0^2)^* = SAT((x_0^1 + 1) S_1)$
3: $S_3 \stackrel{\text{def}}{=} x_3^3 x_2^3 x_1^3 (x_0^3)^* = SAT((x_0^2 + 1) S_2)$
4: $i \leftarrow 3$
5: **repeat**
6: $\quad x_{i+1}^{i+1} x_i^{i+1} x_{i-1}^{i+1} x_1^{i+1} (x_0^{i+1})^* \stackrel{\text{def}}{=} SAT((x_0^i + 1) S_i)$
7: $\quad S_{i+1} \stackrel{\text{def}}{=} x_{i+1}^{i+1} (x_i^{i+1})^* x_1^{i+1} (x_0^{i+1})^*$
8: $\quad i \leftarrow i + 1$
9: **until** $S_i = S_{i-1}$
10: $R \stackrel{\text{def}}{=} (x_1^i + 1)(x_{i-1}^i)^*$
11: **return** S_1, S_i, R

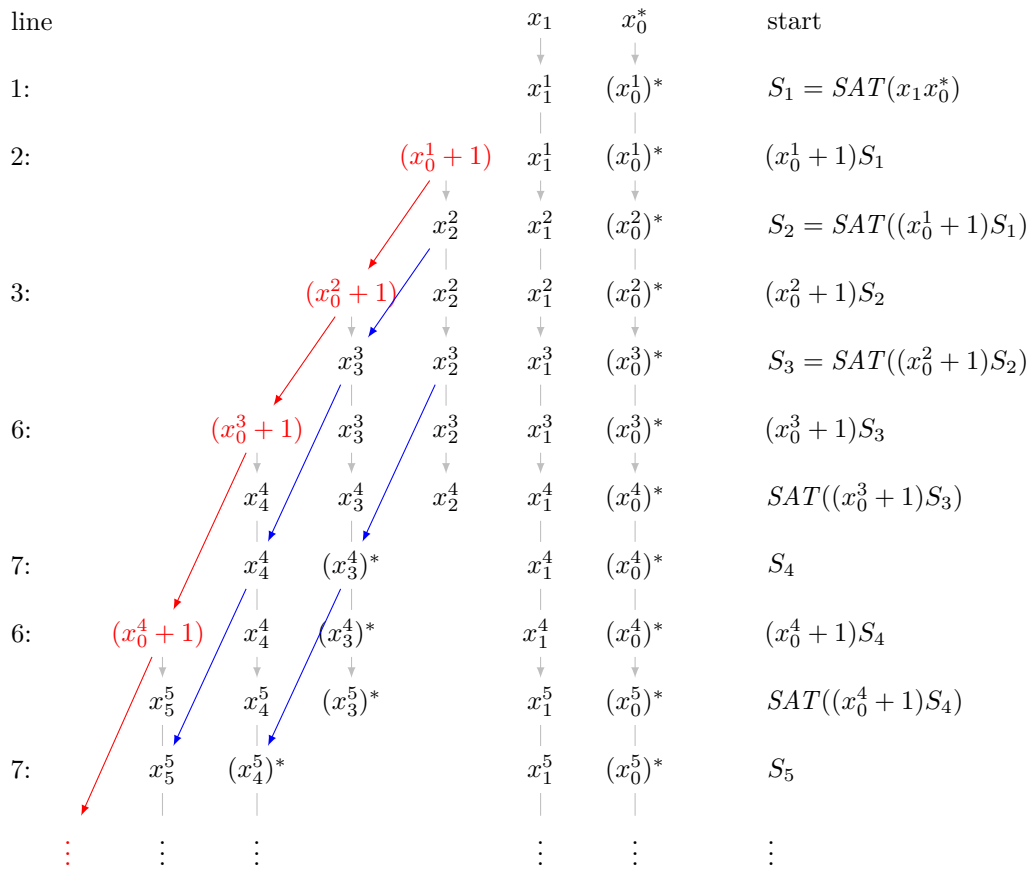


Figure 2 A Run of Algorithm 1 (initial steps). The column on the left indicates the line of code, the middle depicts the current expression and the column on the right recalls its origin. Gray bars indicate that the respective symbols are equal. Arrows denote (set) inclusion between symbols. The gray vertical arrows indicate inclusions due to saturation (Lemma 13), as claimed in item 1 of Lemma 20. Red and blue arrows indicate derived inclusions (as stated in Lemma 20).

Given a length-2 simple expression S_0 where the rightmost symbol is starred, the algorithm will first saturate (Definition 14, in line 1), and then alternatingly rotate a copy of the rightmost symbol (Lemma 17), and saturate the result (see lines 2, 3, 6). Since each such round extends the length of the expression by one, we additionally collapse them (in line 7) by adding an extra Kleene star to the symbol at the second position. The crucial observation for the correctness of this procedure is that the subsumption step in line 7 does not change the cover sets of the respective expressions.

Observe that Algorithm 1 is well defined because the $SAT(S_i)$ are computable by Lemma 13. Termination is guaranteed by the following simple observation.

▶ **Lemma 20.** *Let $x_j^i \in \Sigma$ be the symbols computed by Algorithm 1. Then*
1. *$x_j^{i+1} \supseteq x_j^i$, for all $i > j \geq 0$.*
2. *$x_i^i \supseteq x_{i-1}^{i-1}$ and $x_i^{i+1} \supseteq x_{i-1}^i$, for all $i \geq 3$.*

Proof. The first item is guaranteed by Point 2 of Lemma 13. In particular this means that $x_0^{i+1} \supseteq x_0^i$ and therefore that $(x_0^{i+1}+1) \supseteq (x_0^i+1)$ for all $i \geq 0$ (indicated as red arrows in Figure 2). The second item now follows from this observation by Lemma 15. ◀

▶ **Lemma 21** (Termination). *Algorithm 1 terminates with $i \leq 4 \cdot |P| \cdot (c_{max}+1)$.*

Proof. From Lemma 20 we deduce that for all $i \geq 2$, the expression S_{i+1} is point-wise larger than or equal to S_i with respect to the subset ordering on symbols. The claim now follows from the observation that all expressions $S_{i\geq 3}$ have length 4 and that every symbol $x_i \in \Sigma$ can only increase at most $|P| \cdot (c_{max}+1)$ times. ◀

▶ **Lemma 22** (Correctness). *Suppose that S_1, S_ℓ, R be the expressions computed by Algorithm 1 applied to the simple expression $x_1 x_0^*$. Then $Cover([\![x_1 x_0^*]\!]) = [\![S_1]\!] \cup [\![S_\ell]\!] \cup Cover([\![R]\!])$.*

Proof. Let $S_1, \dots S_\ell$ denote the expressions defined in lines 1,2,3, and 7 of the algorithm. That is, ℓ is the least index i such that $S_{i+1} = S_i$. We define a sequence E_i of expressions inductively, starting with $E_1 \stackrel{\text{def}}{=} S_1$ and if $E_i = e_i^i e_{i-1}^i \dots e_0^i$, we let $E_{i+1} \stackrel{\text{def}}{=} e_{i+1}^{i+1} e_i^{i+1} e_{i-1}^{i+1} \dots e_0^{i+1} \stackrel{\text{def}}{=} SAT((\hat{e}_0^i+1)E_i)$. Here, the superscript indicates the position of a symbol and not iteration. This is the sequence of expressions resulting from unfolding Lemma 19, interleaved with saturation steps, just in line 6 of the algorithm. That is, the expressions E_i are *not* collapsed (line 7) and instead grow in length with i. Still, $E_1 = S_1$, $E_2 = S_2$ and $E_2 = S_3$, but $E_4 \neq S_4$, because the latter is the result of applying the subsumption step of line 7 in our algorithm. Notice that $Cover([\![x_1 x_0^*]\!]) = \left(\bigcup_{k-1 \geq i \geq 1} [\![E_i]\!]\right) \cup Cover([\![E_k]\!])$ holds for all $k \in \mathbb{N}$. We will use that

$$\bigcup_{i\geq 2} [\![E_i]\!] = \bigcup_{i \geq 2} [\![S_i]\!] = [\![S_\ell]\!]. \tag{2}$$

We start by observing that for all $i, j \in \mathbb{N}$ it holds that $e_j^i = x_j^i$. For $i \leq 3$ this holds trivially by definition of $E_i = S_i$. For larger i, this can be seen by induction using Lemma 13. Towards the first equality in Equation (2), let S_i^j be the expression resulting from $S_i = x_i^i (x_{i-1}^i)^* x_1^i (x_0^i)^*$ by unfolding the first star j times. That is, $S_i^j \stackrel{\text{def}}{=} x_i^i (x_{i-1}^i)^{(j)} x_1^i (x_0^i)^*$, where the superscript (j) denotes j-fold concatenation. Clearly, $[\![S_i]\!] = \bigcup_{j\geq 0} [\![S_i^j]\!]$ and so the $\supseteq$-direction of the first equality in Equation (2) follows by

$$\begin{aligned}
[\![S_i^j]\!] = [\![x_i^i (x_{i-1}^i)^{(j)} x_1^i (x_0^i)^*]\!] &\subseteq [\![x_{i+j}^{i+j} \left(x_{i+j-1}^{i+j} x_{i+j-2}^{i+j} \dots x_i^{i+j}\right) x_1^{i+1} (x_0^{i+1})^*]\!] \\
&\subseteq [\![x_{i+j}^{i+j} \left(x_{i+j-1}^{i+j} x_{i+j-2}^{i+j} \dots x_i^{i+j}\right) \left(x_{i-1}^{i+j} \dots x_2^{i+j}\right) x_1^{i+1} (x_0^{i+j})^*]\!] \\
&= [\![E_{i+j}]\!],
\end{aligned}$$

where the first inclusion is due to Lemma 20. The same helps for the other direction:

$$[\![E_i]\!] = [\![x_i^i x_{i-1}^i x_{i-2}^i \dots x_2^i x_1^i x_0^i]\!] \subseteq [\![x_i^i (x_{i-1}^i)^{(i-2)} x_1^i x_0^i]\!] = [\![S_i^{i-2}]\!] = [\![S_i]\!], \tag{3}$$

which completes the proof of the first equality in Equation (2). The second equality holds because $[\![S_i]\!] \subseteq [\![S_{i+1}]\!]$ for all $i \geq 2$, by Lemma 20, and by definition of $S_\ell = S_{\ell+1}$. As a next step we show that

$$\mathit{Cover}([\![S_\ell]\!]) = [\![S_\ell]\!] \cup \mathit{Cover}([\![R]\!]) \tag{4}$$

First observe that $[\![R]\!] = [\![(x_1^\ell + 1)(x_{\ell-1}^\ell)^*]\!] = [\![(x_1^\ell + 1)x_\ell^\ell (x_{\ell-1}^\ell)^*]\!]$ and consequently,

$$\begin{aligned}
\mathit{Cover}([\![R]\!]) &= \mathit{Cover}\Big([\![(x_1^\ell + 1)x_\ell^\ell (x_{\ell-1}^\ell)^*]\!]\Big) \\
&\subseteq \mathit{Cover}\Big([\![x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell]\!]\Big) \\
&\subseteq \mathit{Cover}\Big([\![x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell (x_0^\ell)^*]\!]\Big) = \mathit{Cover}([\![S_\ell]\!])
\end{aligned}$$

where the first equation follows by Corollary 18 and the second because $\mathcal{L}\Big(x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell\Big) \subseteq \mathcal{L}\Big(x_\ell^\ell (x_{\ell-1}^\ell)^* x_1^\ell (x_0^\ell)^*\Big)$. For the left to right inclusion in Equation (4), consider a marking $M \in \mathit{Cover}([\![S_\ell]\!]) \setminus [\![S_\ell]\!]$. We show that $M \in \mathit{Cover}([\![R]\!])$. Recall that $\mathit{Cover}([\![S_\ell]\!])$ consists of all those markings M so that there exists a finite path

$$M_0 \xrightarrow{*}_{Disc} M_0' \xrightarrow{d_1}_{Time} M_1 \xrightarrow{*}_{Disc} M_1' \xrightarrow{d_2}_{Time} M_2 \dots M_{k-1}' \xrightarrow{*}_{Disc} M_k$$

alternating between timed and (sequences of) discrete transition steps, with $M_0 \in [\![S_\ell]\!]$, $M_k \geq M$ and all $d_i \leq \max(\mathit{frac}(M_i'))$.

By our choice of M, there must be a first expression in the sequence which is not a member of $[\![S_\ell]\!]$. Since $[\![SAT(S_\ell)]\!] = [\![S_\ell]\!]$, we can assume an index $i > 0$ so that $M_i \notin [\![S_\ell]\!]$ but $M_{i-1}' \in [\![S_\ell]\!]$ that is, the step that takes us out of $[\![S_\ell]\!]$ is a timed step.

Because $[\![S_\ell]\!] = \bigcup_{i \geq 2} [\![S_i]\!]$, it must hold that $M_{i-1}' \in [\![S_j]\!] = [\![x_j^j (x_{j-1}^j)^* x_1^j (x_0^j)^*]\!]$ for some index $j \geq 2$. We claim that it already holds that

$$M_{i-1}' \in [\![x_j^j (x_{j-1}^j)^* x_1^j]\!]. \tag{5}$$

Suppose not. If $d_i < \max(\mathit{frac}(M_{i-1}'))$ then $M_i \in [\![\emptyset S_j]\!] \subseteq [\![S_j]\!]$ by Lemma 16, contradiction. Otherwise, if $d_i = \max(\mathit{frac}(M_{i-1}'))$, notice that every abstraction $\mathit{abs}_S(M_{i-1}') \in \mathcal{L}(S_j)$ must have $|S| = 4$. So by Lemma 17, $M_i \in [\![(x_0^j + 1)S_j]\!]$. But then again

$$[\![(x_0^j + 1)S_j]\!] \subseteq [\![SAT((x_0^j + 1)S_j)]\!] \subseteq [\![S_{j+1}]\!], \tag{6}$$

contradicting our assumption that $M_i \notin [\![S_\ell]\!]$. Therefore Equation (5) holds. By Lemma 17 we derive that $M_i \in [\![(x_1^j + 1)x_j^j (x_{j-1}^j)^*]\!] = [\![(x_1^j + 1)(x_{j-1}^j)^*]\!] \subseteq [\![(x_1^\ell + 1)(x_{\ell-1}^\ell)^*]\!] = [\![R]\!]$. This concludes the proof of Equation (4).

Notice that by Lemma 19 we have that

$$\mathit{Cover}([\![x_1 x_0^*]\!]) = [\![SAT(x_1 x_0^*)]\!] \cup \mathit{Cover}([\![SAT(x_1 x_0^*)]\!]) = [\![S_1]\!] \cup \mathit{Cover}([\![S_1]\!]). \tag{7}$$

Analogously, we get for every $i \geq 1$ that

$$\mathit{Cover}([\![E_i]\!]) = [\![SAT(E_i)]\!] \cup \mathit{Cover}\big([\![SAT((x_0^i + 1)E_i)]\!]\big) = [\![E_i]\!] \cup \mathit{Cover}([\![E_{i+1}]\!]) \tag{8}$$

This used Lemma 19 and the fact that $SAT(E_i) = E_i$ by construction. Using Equation (8) and that $[\![E_i]\!] \subseteq [\![E_{i+1}]\!]$ for $i \geq 2$, we deduce

$$Cover([\![S_1]\!]) = Cover([\![E_1]\!]) = [\![E_1]\!] \cup \left(\bigcup_{i \geq 2} Cover([\![E_i]\!]) \right). \tag{9}$$

Finally we can conclude the desired result as follows.

$$\begin{aligned} Cover([\![x_1 x_0^*]\!]) &\overset{(7)}{=} [\![S_1]\!] \cup Cover([\![S_1]\!]) \overset{(9)}{=} [\![S_1]\!] \cup Cover\left(\bigcup_{i \geq 2} [\![E_i]\!] \right) \\ &\overset{(2)}{=} [\![S_1]\!] \cup Cover([\![S_\ell]\!]) \\ &\overset{(4)}{=} [\![S_1]\!] \cup [\![S_\ell]\!] \cup Cover([\![R]\!]) \end{aligned}$$

◀

4.3 Main Result

The following theorem summarizes our main claims regarding the ∃COVER problem.

▶ **Theorem 23.** *Consider an instance of ∃COVER with $\mathcal{N} = (P, T, Var, G, Pre, Post)$ a non-consuming TPN where c_{max} is the largest constant appearing in the transition guards G encoded in unary, and let p be an initial place and t be a transition.*

1. *The number of different simple expressions of length m is $B(m) \overset{def}{=} 2^{(|P| \cdot (c_{max}+2) \cdot m) + m}$.*
2. *It is possible to compute a symbolic representation of the set of markings coverable from some marking in the initial set $\mathbb{N} \cdot \{(p,0)\}$, as a finite set of simple expressions. I.e., one can compute simple expressions $S_1, \ldots, S_\ell$ s.t. $\bigcup_{1 \leq i \leq \ell} [\![S_i]\!] = Cover(\mathbb{N} \cdot \{(p,0)\})$ and where $\ell \leq 3 \cdot B(2)$. Each of the S_i has length either 2 or 4.*
3. *Checking if there exists $M \in Cover(\mathbb{N} \cdot \{(p,0)\})$ with $M \longrightarrow_t$ can be done in $\mathcal{O}(|P| \cdot c_{max})$ deterministic space.*

Proof. For Item 1 note that a simple expression is described by a word where some symbols have a Kleene star. There are $|\Sigma|^m$ different words of length m and 2^m possibilities to attach stars to symbols. Since the alphabet is $\Sigma \overset{\text{def}}{=} 2^{P \times [c_{max}+1]}$ and $|[c_{max}+1]| = c_{max}+2$, the result follows.

Towards Item 2, we can assume w.l.o.g. that our TPN is non-consuming by Lemma 8, and thus the region abstraction introduced in Section 4.1 applies. In particular, the initial set of markings $\mathbb{N} \cdot \{(p,0)\}$ is represented exactly by the expression $S_0 \overset{\text{def}}{=} \{(p,0)\}\emptyset^*$ where $\emptyset \in \Sigma$ is the symbol corresponding to the empty set. That is, we have $[\![S_0]\!] = \mathbb{N} \cdot \{(p,0)\}$ and thus $Cover([\![S_0]\!]) = Cover(\mathbb{N} \cdot \{(p,0)\})$.

The claimed expressions S_i are the result of iterating Algorithm 1 until a previously seen expression is revisited. Starting at $i = 0$ and $S_0 \overset{\text{def}}{=} \{(p,0)\}\emptyset^*$, each round will set S_{i+1}, S_{i+2} and S_{i+3} to the result of applying Algorithm 1 to S_i, and increment i to $i+3$.

Notice that then all S_i are simple expressions of length 2 or 4 and that in particular, all expressions with index divisible by 3 are of the form ab^* for $a, b \in \Sigma$. Therefore after at most $B(2)$ iterations, an expression S_ℓ is revisited (with $\ell \leq 3B(2)$). Finally, an induction using Lemma 22 provides that $\bigcup_{1 \leq i \leq \ell} [\![S_i]\!] = Cover(\mathbb{N} \cdot \{(p,0)\})$.

Towards Item 3, we modify the above algorithm for the ∃COVER problem with the sliding window technique. The algorithm is the same as above where instead of recording all the expressions $S_1, \ldots, S_\ell$, we only store the most recent ones and uses them to decide

whether the transition t is enabled. If the index i reaches the maximal value of $3 \cdot B(2)$ we return unsuccessfully.

The bounded index counter uses $\mathcal{O}(\log(B(2)))$ space; Algorithm 1 uses space $\mathcal{O}(\log(B(5)))$ because it stores only simple expressions of length ≤ 5. The space required to store the three expressions resulting from each application of Algorithm 1 is $\mathcal{O}(3 \cdot \log(B(4)))$. For every encountered simple expression we can check in logarithmic space whether the transition t is enabled by some marking in its denotation. Altogether the space used by our new algorithm is bounded by $\mathcal{O}(\log(B(5)))$. By Item 1, this is $\mathcal{O}(|P| \cdot (c_{max} + 2)) = \mathcal{O}(|P| \cdot c_{max})$. ◀

▶ **Corollary 24.** *The ∃COVER problem for TPN is* PSPACE*-complete.*

Proof. The PSPACE lower bound was shown in Theorem 7. The upper bound follows from Lemma 8 and Item 3 of Theorem 23. ◀

5 Conclusion and Future Work

We have shown that *Existential Coverability* (and its dual of universal safety) is PSPACE-complete for TPN with one real-valued clock per token. This implies the same complexity for checking safety of arbitrarily large timed networks without a central controller. The absence of a central controller makes a big difference, since the corresponding problem *with* a central controller is complete for $F_{\omega^{\omega^\omega}}$ [12].

It remains an open question whether these positive results for the controller-less case can be generalized to multiple real-valued clocks per token. In the case *with* a controller, safety becomes undecidable already for two clocks per token [2].

Another question is whether our results can be extended to more general versions of timed Petri nets. In our version, clock values are either inherited, advanced as time passes, or reset to zero. However, other versions of TPN allow the creation of output-tokens with new non-deterministically chosen non-zero clock values, e.g., the timed Petri nets of [3, 4] and the read-arc timed Petri nets of [8].

References

1 Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2):109–127, 2000.

2 Parosh Aziz Abdulla, Johann Deneux, and Pritha Mahata. Multi-clock timed networks. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 345–354, 2004.

3 Parosh Aziz Abdulla, Pritha Mahata, and Richard Mayr. Dense-timed Petri nets: Checking Zenoness, token liveness and boundedness. *Logical Methods in Computer Science*, 3(1), 2007.

4 Parosh Aziz Abdulla and Aletta Nylén. Timed Petri nets and BQOs. In *International Conference on Application and Theory of Petri Nets (ICATPN)*, volume 2075 of *LNCS*, pages 53–70. Springer, 2001.

5 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

6 Benjamin Aminof, Sasha Rubin, Florian Zuleger, and Francesco Spegni. Liveness of parameterized timed networks. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 9135 of *LNCS*, 2015.

7 Rémi Bonnet, Alain Finkel, Serge Haddad, and Fernando Rosa-Velardo. Comparing Petri data nets and timed Petri nets. Technical Report LSV-10-23, LSV Cachan, 2010.

8 Patricia Bouyer, Serge Haddad, and Pierre-Alain Reynier. Timed Petri nets and timed automata: On the discriminating power of Zeno sequences. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 420–431. Springer, 2006.

9 David de Frutos Escrig, Valentín Valero Ruiz, and Olga Marroquín Alonso. Decidability of properties of timed-arc Petri nets. In *International Conference on Application and Theory of Petri Nets (ICATPN)*, volume 1825 of *LNCS*, pages 187–206. Springer, 2000.

10 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001.

11 Eric Goles, Pedro Montealegre, Ville Salo, and Ilkka Törmä. PSPACE-completeness of majority automata networks. *Theoretical Computer Science*, 609(1):118 – 128, 2016.

12 Serge Haddad, Sylvain Schmitz, and Philippe Schnoebelen. The ordinal recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 355–364, 2012.

13 Lasse Jacobsen, Morten Jacobsen, Mikael H. Møller, and Jiří Srba. Verification of timed-arc Petri nets. In *International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 6543 of *LNCS*, pages 46–72, 2011.

14 Ranko Lazić, Tom Newcomb, Joël Ouaknine, A.W. Roscoe, and James Worrell. Nets with tokens which carry data. *Fundamenta Informaticae*, 88(3):251–274, 2008.

15 Valentin Valero Ruiz, Fernando Cuartero Gomez, and David de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *International Workshop on Petri Nets and Performance Models*. IEEE Computer Society, 1999.

16 Jiří Srba. Timed-arc Petri nets vs. networks of timed automata. In *International Conference on Application and Theory of Petri Nets (ICATPN)*, volume 3536 of *LNCS*, pages 385–402. Springer, 2005.